Relational type-checking for MELL proof-structures. Part 1: Multiplicatives

Giulio Guerrieri, Luc Pellissier, Lorenzo Tortora de Falco

Relational type-checking for MELL proof-structures. Part 1: Multiplicatives

Giulio Guerrieri
Dipartimento di Matematica e Fisica, Università Roma Tre, Rome, Italy
Institut de Mathématiques de Marseille, UMR 7373, Aix-Marseille Université, Centrale Marseille
F-13453 Marseille, France
gguerrieri@uniroma.it

Luc Pellissier
LIPN, UMR 7030, Université Paris 13, Sorbonne Paris Cité
F-93430 Villetaneuse, France
luc.pellissier@lipn.univ-paris13.fr

Lorenzo Tortora de Falco
Dipartimento di Matematica e Fisica, Università Roma Tre, Rome, Italy
tortora@uniroma3.it

Relational semantics for linear logic is a form of non-idempotent intersection type system, from which several informations on the execution of a proof-structure can be recovered. An element of the relational interpretation of a proof-structure R with conclusion Γ acts thus as a type (refining Γ) having R as an inhabitant.

We are interested in the following type-checking question: given a proof-structure R, a list of formulæ Γ, and a point x in the relational interpretation of Γ, is x in the interpretation of R? This question is decidable. We present here an algorithm that decides it in time linear in the size of R, if R is a proof-structure in the multiplicative fragment of linear logic. This algorithm can be extended to larger fragments of multiplicative-exponential linear logic containing λ-calculus.

1 Introduction

Intersection types have been introduced as a way of extending the λ-calculus' simple types with finite polymorphism, by adding a new type constructor $\cap$ and new typing rules governing it. A term of type $A \cap B$ can be used in further derivations both as data of type A and as data of type B. Contrarily to simple types (which are sound but incomplete), intersection types present a sound and complete characterization of strong normalization.

Intersection types were first fomulated idempotent, that is, verifying the equation $A \cap A = A$. This corresponds to an interpretation of a typed term $M : A \cap B$ as *M can be used as data of type A or as data of type B*. In a non-idempotent setting (*i.e.* by dropping the equation $A \cap A = A$), the meaning of the typing judgment is strengthened to *M can be used once as data of type A and once as data of type B*. Non-idempotent intersection types have been used to get qualitative and quantitative information on the execution time of λ-terms [1, 3].

Relational semantics is one of the simplest semantics of Linear Logic (LL, [2]). A LL formula is interpreted by a set, and a LL proof-structure [1] by a relation between sets. Relational semantics correspond to a non-idempotent intersection type system, called System R in [1] (see also [8]).

[1]Following [2], we make a difference between proof-structures and proof-nets: a proof-net is a proof-structure corresponding

Submitted to:
ITRS 2016

The relational semantics $\mathbf{Rel}_!$ of the λ-calculus arise from the $*$-autonomous category of sets and relations and the co-monad of finite multisets. Rather than describing its exact structure, we describe the interpretation of simply-typed λ-terms, with base type o.

Let $\mathscr{A}t$ be a set. To each type σ, we associate a set $[\![\sigma]\!]$ as follows:

$$[\![o]\!] = \mathscr{A}t \qquad [\![\sigma \to \tau]\!] = \mathscr{M}_{\mathrm{fin}}([\![\sigma]\!]) \times [\![\tau]\!],$$

where $\mathscr{M}_{\mathrm{fin}}(\cdot)$ denotes the set of finite multisets. We will at times write $X \to \alpha$ as a semantically-flavoured notation for pair $(X, \alpha) \in [\![\sigma \to \tau]\!]$.

To each valid typing sequent $\overline{x} : \overline{\sigma} \vdash M : \tau$ (where $\overline{x} : \overline{\sigma} = x_1 : \sigma_1, \ldots, x_n : \sigma_n$), we associate a set

$$[\![\overline{x} : \overline{\sigma} \vdash M : \tau]\!] \subseteq \mathscr{M}_{\mathrm{fin}}([\![\sigma_1]\!]) \times \cdots \times \mathscr{M}_{\mathrm{fin}}([\![\sigma_n]\!]) \times [\![\tau]\!]$$

as follows:

$$\begin{aligned}
[\![\overline{x} : \overline{\sigma} \vdash x_i : \sigma_i]\!] &= \{(X_1, \ldots, X_n, \alpha) \mid \alpha \in X_i\} \\
[\![\overline{x} : \overline{\sigma} \vdash \lambda y.M : \sigma \to \tau]\!] &= \{(X_1, \ldots, X_n, Y \to \alpha) \mid (X_1, \ldots, X_n, Y, \alpha) \in [\![\overline{x} : \overline{\sigma}, y : \sigma \vdash M : \tau]\!]\} \\
[\![\overline{x} : \overline{\sigma} \vdash MN : \tau]\!] &= \{(X_1, \ldots, X_n, \alpha) \mid \exists Y \in \mathscr{M}_{\mathrm{fin}}([\![\sigma]\!]), (X_1, \ldots, X_n, Y \to \alpha) \in [\![\overline{x} : \overline{\sigma} \vdash M : \sigma \to \tau]\!] \\
&\qquad \forall \beta \in Y, (X_1, \ldots, X_n, \beta) \in [\![\overline{x} : \overline{\sigma} \vdash N : \sigma]\!]\}
\end{aligned}$$

As $\mathbf{Rel}_!$ is a cartesian closed category, if $M =_{\beta\eta} N$, $[\![\overline{x} : \overline{\sigma} \vdash M : \tau]\!] = [\![\overline{x} : \overline{\sigma} \vdash N : \tau]\!]$. We will write $\rhd M : \alpha : \sigma$ for $\alpha \in [\![\vdash M : \sigma]\!]$, emphasizing that the intersection type α refines the simple type σ.

We now give examples of the kind of information that can be recovered from the relational semantics:

- let $\mathbf{B} = o \to o \to o$, the Church encoding of booleans. Let $\mathbf{true} = \lambda xy.x$ and $\mathbf{false} = \lambda xy.y$. Then we have $\rhd\, \mathbf{true} : [*] \to \varnothing \to * : \mathbf{B}$, but not $\rhd\, \mathbf{false} : [*] \to \varnothing \to * : \mathbf{B}$, where $* \in [\![o]\!]$. As a consequence:

  Theorem 1. *Let M be a closed term of type $\mathbf{B}$. Then $M =_{\beta\eta}$ $\mathbf{true}$ if and only if $\rhd M : [*] \to \varnothing \to * : \mathbf{B}$.*

- More generally, a λ-term M has a sort of principal relational type: its 1-point, which can be computed efficiently when M is in normal form. That is, given a term M of type σ in normal form, let M be its 1-point. Then, for any term N of type σ, $M =_{\beta\eta} N$ if and only if $\rhd N : \mathsf{M} : \sigma$.
- Intersection types based on a variant of relational semantics have been shown useful [4] to encode verification problems.
- A variant of relational semantics, Scott semantics, can be used as a faster alternative to β-evaluation [9].
- Given two terms $M_1 : \sigma \to \tau$ and $M_2 : \sigma$ in normal form, it is possible to compute the length of the reduction of $M_1 M_2$ to its normal form [1].

Such information becomes valuable when it is easy to determine whether a point belongs to the relational interpretation of a proof-structure. In other word, we are interested in the tractability of the following decision problem: given a relational element x and a λ-term M, can M be typed by x ?

As the simply-typed λ-calculus embeds in multiplicative-exponential linear logic through the call-by-name translation $o \to o = \,!o \multimap o$, we tackle this study on linear logic proof-nets.

to a derivation in LL sequent calculus. Proof-nets can be characterized among proof-structures via geometric correctness criteria.

(a) Axiom (b) Cut (c) Tensor (d) Par (e) One (f) Bottom

Figure 1: The cells

As a first step towards the resolution of this question, we restrict ourselves to proof-structures in the multiplicative fragment (MLL) of LL. In this particular setting, the interpretation of a formula is finite (up to innocuous renaming). We aim to climb in the ladder of several fragments of multiplicative-exponential LL, providing algorithms of increasing complexity deciding this problem.

This problem has been present since the dawn of LL; indeed, in its seminal article, Girard [2, 3.16. Remark (ii), p. 57] answers the question of the decidability of the following question: given a point x and a proof π, is x in the coherent interpretation[2] of π? The coherent setting is very different from the relational setting; indeed, coherent semantics is not able to distinguish between certain (non-connected) proof-structures [10]. Nonetheless, we note along with him that this problem in the relational setting is trivial for multiplicative proof-structures without cuts: indeed, it suffices to propagate the information present in the conclusions of a proof-structure. Cuts allow to hide certain parts of the proof-structure from its conclusions (see Figure 4); in the presence of cuts, cycles need a special treatment. In this article, we will introduce a general framework deciding this problem for multiplicative proof-structures with cuts, and explain how it can be adapted to larger fragments of LL containing the λ-calculus.

We will define a variant of Vector Addition Systems (VAS, see [7]) that encode naturally our decision problem. The machine bears a close resemblance with the Interaction Abstract Machine (see for instance [6]). It has indeed been known for a long time in the Linear Logic community that Geometry of Interaction and relational semantics enjoy a certain closeness. This work aims to bridge them on the operational side.

We provide an algorithm that decides in time linear in the size of the multiplicative proof-structure R whether a point x is in the relational interpretation of R. We give indications on how this algorithm can be extended in a bilinear (in the size of the term and of the point) algorithm in the case of λ-terms.

2 Elements of MLL syntax

The set of MLL *formulas* is generated by the grammar:

$$A,B,C ::= X \mid X^{\perp} \mid 1 \mid \perp \mid A \otimes B \mid A \parr B\,.$$

where X ranges over an infinite countable set of *propositional variables*. The linear negation $A^{\perp}$ of a formula A is involutive, *i.e.* $A^{\perp\perp} = A$, and defined via De Morgan laws $1^{\perp} = \perp$ and $(A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp}$. If $\Gamma = (A_1, \ldots, A_n)$ is a finite sequence of MLL formulas (with $n \in \mathbf{N}$), then $\parr\Gamma = A_1 \parr \cdots \parr A_n$; in particular, if $n = 0$ then $\parr\Gamma = \perp$.

Proof-structures offer a syntax for a graphical representation of MLL proofs. MLL proof-structures are directed labelled graphs Φ built from the *cells* defined in Figure 1. We call *ports* the directed edges of such graphs, labelled by MLL formulas. For every cell, its ports are divided into *principal ports* (outgoing in the cell, depicted down in the picture) and *auxiliary ports* (incoming in the cell, depicted up).

[2]The coherent semantics of linear logic being the one closest to the dynamics of cut-elimination. Relational semantics can be seen as a simplification of it.

Let Φ be a MLL proof-structure. We denote by $\mathscr{P}(\Phi)$ the set of its ports and $\mathscr{C}(\Phi)$ the set of its cells. Let c be a cell. We denote by:

- $\mathsf{tp}_\Phi(c)$ the type of c, ranging in $\{1, \bot, \otimes, \parr, ax, cut\}$;
- $\mathsf{P}^{\mathsf{pri}}_\Phi(c)$ the principal ports of c. It is either
  - a port, if c is of type $1, \bot, \otimes, \parr$;
  - an ordered pair of ports $\langle p_1, p_2\rangle$, if c is of type ax;
  - empty, if c is of type cut;
- $\mathsf{P}^{\mathsf{aux}}_\Phi(c)$ the auxiliary ports of c. It is either
  - empty, if c is of type $ax, 1, \bot$;
  - an ordered pair of ports $\langle p_1, p_2\rangle$, if c is of type cut, $\parr$ or $\bot$.

In a MLL proof-structure Φ, any port that is principal for some cell but is not auxiliary for any cell of Φ is called a *conclusion* of Φ. We will only consider in the sequel MLL proof-structures with a fixed (total) order on their conclusions. Given a list of MLL formulæ $\Gamma = (A_1, \ldots, A_n)$ with $n \in \mathbf{N}$, we say that an MLL proof-structure Φ is of *conclusion* Γ if the conclusions of Φ are the ordered sequence $p_1 < \cdots < p_n$ of ports of Φ and $\mathsf{tp}_\Phi(p_i) = A_i$ for every $i \in \{1, \ldots, n\}$.

The MLL proof-structure of fig. 2a has two conclusions, one on the far right, the other on the far left. We will always depict proof-structures with conclusions ordered from left to right, so R is of conclusion $(A \otimes B, A^\perp \parr B^\perp)$.

3 Elements of relational semantics

We introduce here a variant of relational semantics (the simplest semantics of Linear Logic, where formulæ are interpreted by sets and proof-structures as relations between sets) parametrized by a set $\mathscr{V}$ of variables. In this variant, parts of a relational element can be left uninterpreted, allowing for unification. We will use this feature in section Section 4.

Definition 1 (Web of a MLL formula)**.** Let $\mathscr{A}t$ be a countably infinite set that doesn't contain the symbols of MLL or the empty sequence $()$; the elements of $\mathscr{A}t$ are called *atoms*.

Let $\mathscr{V}$ be a set disjoint from $\mathscr{A}t$ whose elements are the *atomic variables*.

By induction, we define a function $|\cdot|_\mathscr{V}$ on MLL formulæ by:

$$\begin{aligned}|X^\perp|_\mathscr{V} = |X|_\mathscr{V} &= \mathscr{A}t \cup \mathscr{V}, \text{ for all propositional variable } X;\\ |1|_\mathscr{V} = |\bot|_\mathscr{V} &= \{()\};\\ |A \otimes B|_\mathscr{V} = |A \parr B|_\mathscr{V} &= (|A|_\mathscr{V} \times |B|_\mathscr{V}) \cup \mathscr{V},\end{aligned}$$

For a formula A, the set $|A|_\mathscr{V}$ is called the *web of* A, whose elements are the *points of* A.

We write $\mathbb{REL} = \bigcup_A |A|_\mathscr{V}$ the relational universe, where A range over all MLL formulæ.

The usual relational web of a MLL formula is recovered as $|\cdot|_\varnothing$. We fix from now on an infinite set $\mathscr{V}$ of variables. Note that, for any MLL formula A, one has $|A|_\mathscr{V} = |A^\perp|_\mathscr{V}$.

We define relational experiments straightforwardly on multiplicative proof-structures by adapting the definition in [2]. Let ρ be a MLL proof-structure. A partial experiment of ρ is a partial function of the ports of ρ associating with a port a relational element coherently with the structure of ρ.

Definition 2 (Experiment of a MLL proof-structure)**.** Let Φ be a MLL-ps.

A *partial experiment* e of Φ is a partial function associating with $p \in \mathscr{P}(\Phi)$ an element of $|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}$ verifying the following conditions, if e is defined on all the mentioned ports: let c be a cell in $\mathscr{C}(\Phi)$,

- if c is of type *ax* with $\mathsf{P}^{\mathsf{pri}}_{\Phi}(c) = \langle p, q\rangle$, then $\mathsf{e}(p) = \mathsf{e}(q)$;
- if c is of type *cut* with $\mathsf{P}^{\mathsf{aux}}_{\Phi}(c) = \langle p, q\rangle$, then $\mathsf{e}(p) = \mathsf{e}(q)$;
- if c is of type 1 or $\perp$ with $\mathsf{P}^{\mathsf{pri}}_{\Phi}(c) = q$, then $\mathsf{e}(q) = ()$;
- if c is of type $\otimes$ or $\parr$ with $\mathsf{P}^{\mathsf{aux}}_{\Phi}(l) = \langle p_1, p_2\rangle$, $\mathsf{P}^{\mathsf{pri}}_{\Phi}(l) = q$, then $\mathsf{e}(q) = (\mathsf{e}(p_1), \mathsf{e}(p_2))$.

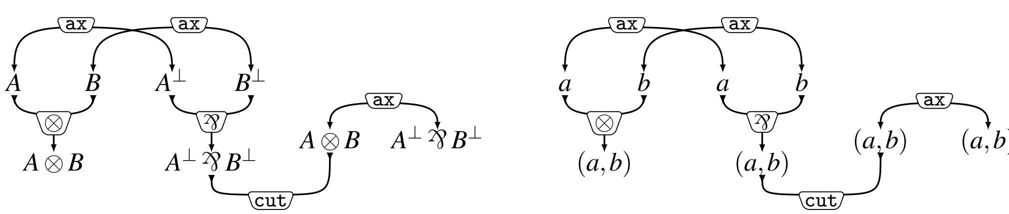
An *experiment* is a partial experiment defined on all ports, whose codomain can be restricted to $\bigcup_{p\in\mathscr{P}(\Phi)} |\mathsf{tp}_\Phi(p)|_{\varnothing}$.

If we consider the Cartesian product of sets and relations to be literally associative, an experiment of a proof-structure of type $\Gamma = (A_1, \ldots, A_n)$ defines naturally an element of $|\parr\Gamma|_{\mathscr{V}}$, its *result*. We write $|\mathsf{e}|$ the result of an experiment e.

The relational interpretation of a MLL proof-structure Φ is then $[\![\Phi]\!] = \{|\mathsf{e}| : \mathsf{e} \text{ experiment of } \Phi\}$.

If we see the relational semantics as a non-idempotent intersection type system, an experiment of a MLL proof-structure is a type derivation, and its result is the conclusion of this type derivation. Just as we did for the λ-calculus in the introduction, we write $\rhd R : \alpha : \Gamma$ for $\alpha \in [\![R]\!] \subseteq |\parr\Gamma|_{\mathscr{V}}$. The point α acts both as a witness of the fact that Γ types R, while refining this type.

The function depicted in Figure 2b is an experiment e of the proof-structure of Figure 2a, where a and b are atoms. The experiment e proves $\rhd R : ((a,b),(a,b)) : (A \otimes B) \parr (A^\perp \parr B^\perp)$.



(a) The MLL proof-structure R

(b) The experiment e of R

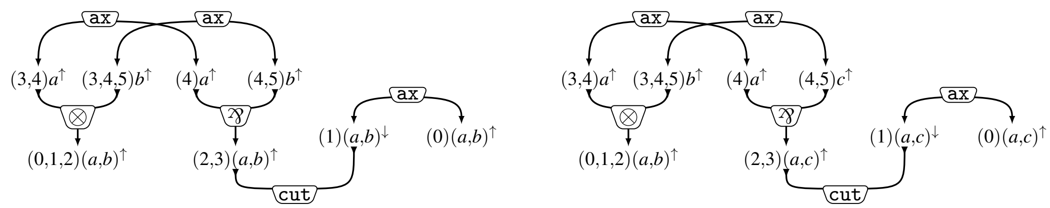
Figure 2: A MLL-proof structure and an experiment on it

4 Semantic typing

We will now describe the main idea graphically, on two examples. In Figure 3, we try, starting from the conclusions, to build an experiment of R, one (Figure 3a) with putative result $((a,b),(a,b))$, the other (Figure 3b) with putative result $((a,b),(a,c))$ (with $b \neq c$). Tokens travel through the proof-structure, encapsulating an element of the relational interpretation of the port they are sitting on. We depict the tokens at different step (where each step is defined by one token moving). Each token is depicted as its travel direction, its content, and the step on which the token exist: $(3,4)a^\uparrow$ meaning that a token is there on steps 3 and 4, containing a and going up.

Let's describe possible execution steps in Figure 3a:

0. two upward tokens containing (a,b) are placed on each conclusion (the principal port of the $\otimes$-cell and the right principal port of the right axiom);



(a) A succesfull typing (b) A failing typing

Figure 3: Two typings or R

1. the right token goes up through the right axiom, and exits downwards from its left principal port;
2. the same token goes down through the cut and exits upwards from its left auxiliary port;
3. the other token (on the principal port of the $\otimes$-cell) gets split in two upwards tokens, one containing a on the left auxiliary port of the $\otimes$-cell, the other containing b on its right auxiliary port;
4. the right token containing (a,b) on the principal port of the $⅋$-cell gets split in two upwards tokens, one containing a, the other containing b;
5. the right token containing a goes up through an axiom, exits downwards from its other principal port and meets an upwards token containing a too. They annihilate each other;
6. the right token containing b goes up through an axiom, exits downwards from its other principal port and meets an upwards token containing b too. They annihilate each other.

As there are no more tokens on the proof-structure, we say that the execution is successful, and so we proved the judgment $\rhd R : ((a,b),(a,b)) : (A \otimes B, A^\perp ⅋ B^\perp)$. Conversely, the same thing happens in Figure 3b, apart from the last step:

6. the right token containing b goes up through an axiom, exits downwards from its other principal port and meets an upwards token containing c. Nothing happens, the machine is stuck.

As the machine is stuck with tokens on it, we say that the execution has failed, and we proved the negation of the judgment $\rhd R : ((a,b),(a,c)) : (A \otimes B, A^\perp ⅋ B^\perp)$.

We will formalize this mechanism in Section 6.

We want our algorithm to be able to handle cases like the proof-structure in Figure 4, where part of the information carried by an experiment can not be retrieved from its conclusion.

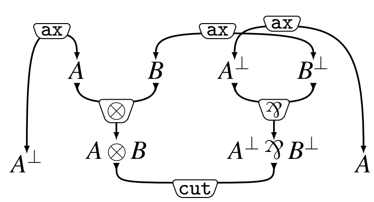


Figure 4: A cyclic proof-structure R

By simply propagating a relational point using the same strategy as defined before, we cannot guess which relational element should be the image of the port of type B through the experiment. In a way, all the information in the cycle concerning B is hidden from the conclusions. We solve this problem by introducing variables (which we already forced into the definition of the relational web of a formula): some transitions can be fired with incomplete information which will be checked later.

5 Vector Addition Systems

Vector Addition Systems (VAS) (for instance, [7]) are a prominent class of infinite state systems. They comprise a finite number of counters ranging over the (non-negative) natural numbers. When taking a transition, an integer can be added to a counter, provided it stays positive. To give an example, let us consider a VAS with two counters. Let α be defined as the displacement $(1,-1)$. The transition $(1,2) \xrightarrow{\alpha} (2,1)$ is valid, while α does not define any transition starting from the state $(1,0)$, because $0+(-1)$ is negative.

VAS are particularly well-suited to represent systems with an infinite number of states. Our idea here is to encode the presence, direction and content of a token in a variant of VAS: to each port p is associated a counter, which is set to 0 if there is no token on p, a if there is an upwards token containing the relational element a on p, $-a$ if there is a downwards token containing a on p. While the systems studied in the sequel have an essentially finitary behaviour, it is not the case in extensions to the exponentials of linear logic: the relational type-checking of MELL-proof structures with exponentials will imply the presence of any arbitrary number of tokens on a given port.

All counters in VAS are natural numbers. We depart from traditional VASs for two reasons: we want to be able to encode tokens going up, but also going down. We also want counters to account for tokens containing a relational element, so the counters are to be taken is a space engendered by relational elements with coefficients. For multiplicative proof-structures, we can restrict ourselves to the case where every coefficient is in $\mathbb{B}_3 = \{-1,0,1\}$. As such, the counter associated to a port $p \in \mathscr{P}(\Phi)$ is a formal series with coefficients in $\{-1,+1\}$. We denote by $\mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}]$ the set of such formal series. It is endowed with a partial sum and a partial difference (by extending pointwise the partial sum and partial difference of $\mathbb{B}_3$, seen as a subset of $\mathbf{Z}$). The configuration of the machine ought then to be an element of the dependent product $\prod_{p\in\mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}]$, which we will write as a function.

6 The relational interaction abstract machine

We are now ready to give the formal definition of the Relational Interaction Abstract Machine for MLL, which decides semantic typing judgments.

Given a relation $R \subseteq A \times B$ and any $a \in A$ and $b \in B$, $a\,R\,b$ stands for $(a,b) \in R$.

Definition 3 (Relational Interaction Abstract Machine for MLL)**.** Let Φ be a MLL proof-structure.

An *environment* is a finite map from $\mathscr{V}$ to $\mathbb{REL}$. The set of environments of Φ is denoted by $\mathfrak{Env}$.

The *relational interaction abstract machine for* MLL (RIAM) associated with Φ, denoted by M^Φ, has the following components:

- its alphabet is $\Sigma = \mathfrak{Env} \cup \mathscr{C}(\Phi)$, where $\mathscr{C}(\Phi)$ is the set of cells of Φ;
- its set of configurations is $\prod_{p\in\mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}]$.

The RIAM associated with Φ has two kinds of transitions: the *displacement transitions* or the *unification transitions*.

The displacement transitions are labelled by an element c of $\mathscr{C}(\Phi)$. The binary relation $\xrightarrow{c}$ on configurations of M^Φ is defined by:

$$x \xrightarrow{c} x' \ \text{ if } c\,\delta\,(x - x').$$

where the *displacement relation* $\delta \subseteq \mathscr{C}(\Phi) \times \prod_{p\in\mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_\Phi(p)|_{\mathscr{V}}]$ is the relation defined by:

- if c is of type ax, let $\mathsf{P}^{\mathsf{pri}}_{\Phi}(c) = \langle p,q\rangle$ and $\forall a \in |\mathsf{tp}_{\Phi}(p)|_{\mathscr{V}}, c\,\delta \begin{cases} p & \mapsto & -a \\ q & \mapsto & -a \\ r & \mapsto & 0, \text{ if } r \neq p,q \end{cases}$

- if c is of type cut, let $\mathsf{P}^{\mathsf{aux}}_{\Phi}(c) = \langle p,q\rangle$ and $\forall a \in |\mathsf{tp}_{\Phi}(p)|_{\mathscr{V}}, c\,\delta \begin{cases} p & \mapsto & a \\ q & \mapsto & a \\ r & \mapsto & 0, \text{ if } r \neq p,q \end{cases}$

- if c is of type 1 or $\perp$, let $\mathsf{P}^{\mathsf{pri}}_{\Phi}(c) = p$ and $c\,\delta \begin{cases} p & \mapsto & -() \\ r & \mapsto & 0, \text{ if } r \neq p \end{cases}$

- if c is of type $\otimes$ or $⅋$, let $\mathsf{P}^{\mathsf{pri}}_{\Phi} = q$ and $\mathsf{P}^{\mathsf{aux}}_{\Phi} = \langle p_1, p_2\rangle$, and $c\,\delta \begin{cases} p_1 & \mapsto & \bullet \\ p_2 & \mapsto & \circ \\ q & \mapsto & -(\bullet,\circ) \\ r & \mapsto & 0, \text{ if } r \neq p_1,p_2,q \end{cases}$.

  where $\bullet, \circ \in \mathscr{V}$ are two fresh variables.

The unification transitions are labelled by environments and defined on couples of configurations by:

$$x \xrightarrow[\mathbf{u}]{s} s(x) \text{ if } \exists p \in \mathscr{P}(\Phi), \begin{cases} x(p) = a_1 - a_2 + \vec{a},\ a_1 \neq a_2,\ \vec{a} \in \mathbb{B}_3[|\mathsf{tp}_{\Phi}(p)|_{\mathscr{V}}] \\ s = \text{m.g.u.}(a_1, a_2) \end{cases}$$

where m.g.u. denotes the most general unifier. Such a transition unifies (and so annihilates) two elements of a formal sum of opposite sign in one of the counters.

We define $\xrightarrow{\sigma}$, for $\sigma \in \Sigma^\star$ by relational composition: let $\sigma = a_1 s_2 \cdots a_{n-1} s_n$, we set

$$x \xrightarrow{\sigma} x' \text{ if } \exists (x_i)_{1 \leqslant i < n}, x \xrightarrow{a_1} x_1 \xrightarrow[\mathbf{u}]{s_2} \cdots \xrightarrow{a_{n-1}} x_{n-1} \xrightarrow[\mathbf{u}]{s_n} x.$$

We say that σ is an *execution* of M^Φ.

We denote by $\underset{\Phi}{\rightarrow}$ the *reachability binary relation* defined by $x \underset{\Phi}{\rightarrow} x'$ if $\exists \sigma \in \Sigma^\star, x \xrightarrow{\sigma} x'$.

We say that an element $x \in \prod_{l \in \mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_{\Phi}(l)|_{\mathscr{V}}]$ is *accepted* or *recognized* by M^Φ if $x \underset{\Phi}{\rightarrow} 0$. and that a $\sigma \in \Sigma^\star$ accepts (or recognizes) x if $x \xrightarrow{\sigma} 0$.

If $x = (x_1, \ldots, x_n) \in |⅋\Gamma|_{\mathscr{V}}$, where $\Gamma = (A_1, \ldots, A_n)$ and the MLL proof-structure Φ has n conclusions $p_1 < p_2 < \cdots < p_n$, we say that M^Φ *accepts* x if it accepts the element of $\prod_{p \in \mathscr{P}(\Phi)} \mathbb{B}_3[|\mathsf{tp}_{\Phi}(p)|_{\mathscr{V}}]$ associating 0 with every port which is not a conclusion, and x_i to p_i.

We will describe the RIAM associated to the proof-structure Φ in Figure 5. Its ports are numbered, its cells are named $\{\mathtt{ax}_{1,2}, \mathtt{ax}_{4,5}, \otimes\}$.

The RIAM associated to Φ has $\Sigma = \mathfrak{Env} \cup \{\mathtt{ax}_{1,2}, \mathtt{ax}_{4,5}, \otimes\}$ as alphabet, $|A|_{\mathscr{V}} \times |A|_{\mathscr{V}} \times |A \otimes B|_{\mathscr{V}} \times |B|_{\mathscr{V}} \times |B|_{\mathscr{V}}$ as set of configurations and its displacement relation δ is defined by, for $\circ, \bullet$ fresh variables:

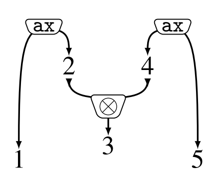


Figure 5: A named proof-structure Φ

$$\begin{aligned} \forall a \in |A|_{\mathscr{V}}, \mathtt{ax}_{1,2}\,\delta\,&(-a,-a,0,0,0) \\ \forall b \in |B|_{\mathscr{V}}, \mathtt{ax}_{4,5}\,\delta\,&(0,0,0,-a,-a) \\ \otimes\,\delta\,&(0,\circ,-(\circ,\bullet),\bullet,0). \end{aligned}$$

The relational element $(a,(a,b),b)$ is recognized by the word $\mathtt{ax}_{1,2}\otimes\{\bullet \mapsto b, \circ \mapsto a\}\mathtt{ax}_{4,5}$.

7 Recognition of the relational interpretation

Among all executions of a RIAM, some are in normal form: intuitively, they don't create any tokens but only propagate those already present in the initial configuration.

Definition 4 (Normal execution)**.** Let Φ be a MLL proof-structure. An execution σ of M^Φ is *normal* if

- for each displacement transition $x \xrightarrow{c} x'$ in σ, there exists a $p \in \mathscr{P}(\Phi)$ such that $x(p) \neq 0$ and $x'(p) = 0$;
- each displacement transition $x \xrightarrow{c} x'$ in σ such that there exists $p \in \mathscr{P}(\Phi)$ such that $x'(p) = -a + a'$ is followed by an unification transition $x' \xrightarrow[\mathbf{u}]{s} x''$ such that $x''(p) = 0$.

Lemma 2. *Let Φ be a MLL proof-structure of conclusion Γ. Let $x \in |\parr\Gamma|_\varnothing$.*
A normal successful run of M^Φ on x is at most of length twice the number of cells of Φ.

Lemma 3. *Let Φ be a MLL proof-structure with conclusion Γ. Let $x \in |\parr\Gamma|_\varnothing$ be such that M^Φ accepts x. Then, there exists a normal execution of M^Φ that recognizes x.*

Normal executions of the machine can be used to define a partial experiment on all ports connected to the conclusions, while an experiment can be sequentialized in an execution.

Theorem 4. *Let Φ be a MLL proof-structure with conclusions Γ (a list of MLL formulæ). Let $x \in |\parr\Gamma|_\varnothing$. Then, $x \in [\![\Phi]\!]$ if and only if there exists a (normal) execution of M^Φ recognizing x.*

As moreover, a run with a counter in a state of the form $a + b$, with a and b not unifiable cannot be extended in a successful run, we get:

Corollary 5. *The RIAM decides judgements of the form $\rhd \Phi : x : \Gamma$ in time $O(|\mathscr{C}(\Phi)|)$ (i.e. in time linear in the size of Φ) by attempting a normal run which sequence of displacement transition refine the tree order of cells in Φ.*

8 Extending to the lambda-calculus

The techniques in this article can be extended to the λ-calculus translated in MELL proof-structure through the call-by-name translation $\alpha \to \beta = !\alpha \multimap \beta$. The formulæ of MELL are those of MLL to which is added two new connectives, ! and ?. The syntax of the proof-structure is enriched with a !-cell, having an arbitrary number of unordered inputs, and a box constructor, taking a proof-structure and encapsulating.

The interpretation of the exponential cells are a bit tedious to define. An elegant way is described in [5]. It is multi-set based: the interpretation of a formula $!A$ or $?A$ is a multiset. The interpretation of a box is the multiset containing multiple interpretations of the content of the box.

We extend the definition of the abstract machine to handle the exponentials:

- coefficients of the machine have to be taken in $\mathbf{Z}$ (and no more in $\mathbb{B}_3$), moreover on top of containing a relational element, tokens contain also a stack of timestamps remembering when boxes are entered;
- new rules must be added, allowing to pass through exponential cells. They amount to splitting the content of multi-cells in all possible ways;

Together with an appropriate definition of normal run, this allows to prove:

Theorem 6. *The RIAM for the λ-calculus decides judgements of the form $\rhd M : x : \Gamma$ in time $O(|M| \times |x|)$.*

As a corollary of this Theorem and of Theorem 1, we get the following (unpublished) result:

Corollary 7 (Terui, 2012)**.** *Let* $\mathbf{W} := (o \to o) \to (o \to o) \to o \to o$ *be the Church encoding of binary strings.*

Let M be a closed λ-term of type $\mathbf{W} \to \mathbf{B}$. *It decides a language* $\mathscr{L}$.

$\mathscr{L}$ *is in* **LinTIME** *(deterministic linear time).*

The proof consisting of checking, for an encoded string $s : \mathbf{W}$, whether $\rhd Ms : [*] \to \varnothing \to * : \mathbf{B}$, which is done in time linear in the size of the translation of M, itself linear in the size of M.

The result is surprising, as simply-typed λ-terms of type $\mathbf{N} \to \mathbf{N}$ (where $\mathbf{N}$ is the Church encoding of natural numbers) can represent a function of complexity an arbitrary tower of exponentials.

Acknowledgements

Work partially supported by ANR projects COQUAS ANR-12-JS02-006-01 and ELICA ANR-14-CE25-0005. The authors thank Damiano Mazza for pointing to them the last Corollary.

References

[1] Daniel de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. To appear in *Mathematical Structures in Computer Science*, 2009. Available at `http://arxiv.org/abs/0905.4251`.

[2] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[3] Stéphane Graham-Lengrand and Alexis Bernadet. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, Volume 9, Issue 4(4), October 2013.

[4] Charles Grellois and Paul-André Melliès. Relational semantics of linear logic and higher-order model checking. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, pages 260–276, 2015.

[5] G. Guerrieri, L. Pellissier, and L. Tortora de Falco. Syntax, Taylor expansion and relational semantics of MELL proof-structures: an unusual approach. Technical report, 2016. Available at `http://logica.uniroma3.it/~tortora/mell.pdf`.

[6] Olivier Laurent. A Token Machine for Full Geometry of Interaction (Extended Abstract). In Samson Abramsky, editor, *Typed Lambda Calculi and Applications '01*, volume 2044 of *Lecture Notes in Computer Science*, pages 283–297. Springer, May 2001.

[7] Jérôme Leroux. The general vector addition system reachability problem by Presburger inductive invariants. *Logic In Computer Science*, 2009.

[8] Luca Paolini, Mauro Piccolo, and Simona Ronchi Della Rocca. Essential and relational models. *Mathematical Structures in Computer Science*, FirstView:1–25, September 2015.

[9] Kazushige Terui. Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, pages 323–338, 2012.

[10] Lorenzo Tortora de Falco. Obsessional Experiments For Linear Logic Proof-Nets. *Mathematical Structures in Computer Science*, 13(6):799–855, December 2003.